ERB-964

NRCC No. 23031

National Research Council Canada
Conseil national de recherches Canada

Division of Electrical Engineering
Division de génie électrique

CHORUS - A MULTIPROCESSOR ARCHITECTURE FOR REAL-TIME CONTROL APPLICATIONS

D. GREEN

DECEMBER 1983

Canada

Copyright 1983 by
National Research Council of Canada

Permission is granted to quote short excerpts and to reproduce figures and tables from this report, provided that the source of such material is fully acknowledged.

Additional copies are available free of charge from:

Editorial Office, Room 301
Division of Electrical Engineering
National Research Council of Canada
Ottawa, Ontario, Canada
K1A 0R8

Copyright 1983 par
Conseil national de recherches du Canada

Il est permis de citer de courts extraits et de reproduire des figures ou tableaux du présent rapport, à condition d'en identifier clairement la source.

Des exemplaires supplémentaires gratuits peuvent être obtenus à l'adresse suivante:

Bureau des publications, Pièce 301
Division de génie électrique
Conseil national de recherches du Canada
Ottawa (Ontario) Canada
K1A 0R8

CHORUS - A MULTIPROCESSOR ARCHITECTURE FOR REAL-TIME CONTROL APPLICATIONS

D. Green

ERB-964
NRCC No. 23031

DECEMBER 1983

TABLE OF CONTENTS

		Page
ABSTRACT/RÉSUMÉ		v
CHAPTER 1.	INTRODUCTION	1
CHAPTER 2.	AN OVERVIEW OF THE HARDWARE CONFIGURATION OF CHORUS	2
CHAPTER 3.	THE MAILBOX AND INTERPROCESSOR INTERRUPT GENERATOR (MIIG)	4
	Introduction	4
	Message Passing with the MIIG	5
	Why a Separate Mailbox is Required	5
	Detailed Hardware Description	6
CHAPTER 4.	BUS AND CHASSIS CONFIGURATION DETAILS	14
CHAPTER 5.	CONCLUSIONS	15
REFERENCES		16

LIST OF FIGURES

	Page
1 Memory and I/O Configuration	3
2 Generalized Chorus Configuration	4
3 A software view of the mailbox and interprocessor interrupt generator	6
4 Address Map of the MIIG	8
5 MIIG design - IEEE 796 bus interface and control logic	10
6 MIIG design - mailbox and data paths	11
7 MIIG design - interprocessor interrupt generator	12
8 Mailbox timing	13
9 Interrupt generator timing	14
10 Interconnection of interprocessor interrupts on the backplane	15

ABSTRACT

The design of a multiprocessor architecture for real-time control is described. This system, called Chorus, is the first implementation of a system that runs under the Harmony multitasking operating system. It can easily be expanded to include up to sixteen 68000-based single board computers. Harmony relies on a message-passing mechanism for intertask communication and synchronization. The hardware that was developed to implement this mechanism is described in detail. It consists of a mailbox and an interprocessor interrupt generator. Chorus has been demonstrated in the control of a robot arm that uses vision sensing for identifying and locating randomly oriented parts.

RÉSUMÉ

L'auteur décrit les caractéristiques de l'architecture d'un multi-processeur pour la commande en temps réel. Le système, désigné Chorus, est la première réalisation d'un dispositif commandé par le système d'exploitation multi-tâches Harmony. Il peut facilement être élargi de manière à contenir jusqu'à 16 ordinateurs à carte unique basés sur un 68000. Harmony compte sur un mécanisme de transmission des messages pour la communication et la synchronisation inter-tâches. Le matériel mis au point pour la réalisation de ce mécanisme est décrit en détail. Il comprend une zone de communication et un générateur d'interruptions interprocesseurs. Chorus a fait l'objet d'une démonstration pour la commande d'un bras robot qui se sert de détecteurs visuels pour l'identification et la localisation de pièces disposées au hasard.

CHORUS - A MULTIPROCESSOR ARCHITECTURE FOR REAL-TIME CONTROL APPLICATION

D. Green

CHAPTER 1. INTRODUCTION

Over the last few years, developments in silicon technology and design methodology have resulted in the commercial production of advanced micro-electronic components which are available at very reasonable cost. In turn, these have been exploited in the design of advanced microcomputer systems. Chorus is an example of such a system.

Chorus is a multiprocessor architecture that has been designed specifically for real-time control applications. Its initial application has been to control a robot arm equipped with vision sensing. Chorus is the first implementation of a system intended to run under the Harmony operating system (1). Together they form a multitasking, multiprocessor system that is well suited for the asynchronous control environment.

As the complexity of an application increases, for example, the addition of one or more robot arms and associated sensors, the processing capacity of Chorus can be increased correspondingly. In its current configuration, Chorus consists of two Motorola 68000-based single board microcomputers. However, the system can be readily expanded to include up to 16 such processors. Physically this merely involves plugging the additional processors into the card-cage. Similarly, the Harmony operating system was designed to conform to this modular structure. Transferring tasks from processor to processor at compile time has little effect upon the program structure as seen by the user. Stated another way, as the complexity of the application increases so too does the degree of concurrency in the Chorus machine.

Currently program development for Chorus is done on a VAX-11/780. The core images are downloaded to Chorus via a serial interface. A separate core image is required for each processor and these are downloaded in turn. As Chorus expands, additional core images are required. These are compiled and linked separately. Consequently the burden upon the user resulting from the expansion of Chorus is almost insignificant.

This report is concerned with the hardware implementation of Chorus. Chapter 2 provides a general overview of the hardware. Chapter 3 describes in detail the "Mailbox and Interprocessor Interrupt Generator" (MIIG) board that was designed to support the intertask/interprocessor message-passing mechanism. Finally Chapter 4 outlines the configuration of the IEEE 796 bus ("Multibus") (Multibus is a trademark of Intel Corporation) that has been used in this implementation.

CHAPTER 2. AN OVERVIEW OF THE HARDWARE CONFIGURATION OF CHORUS

While the Harmony operating system is essentially processor and bus independent, the Chorus system has been implemented on the IEEE 796 bus. Probably the most important consideration in choosing this bus was the large number of vendors that supply hardware for it (2). This number continues to grow. Apart from selecting a single board computer with the appropriate dual-port memory configuration it was also easy to locate suppliers of analog I/O boards, video frame grabbers, card extenders and card-cages that were required. The IEEE 796 bus specification insists that all boards designed for the bus must clearly show the compliance level of that board. This facilitated the selection of board-level components.

The single board computer used in Chorus is the OB68K1A board manufactured by Omnibyte (3). This board was chosen (Sept. 1982) for several reasons. It employs a 10 MHz 68000 microprocessor and contains 128 kbytes of 120 ns RAM that can be accessed locally with no wait states. In the future, when the 256 kbit RAM's become available, it will be possible to upgrade the board to 512 kbytes. The board has two parallel ports permitting 16-bit parallel I/O, and has two serial ports and an onboard timer as well. Its most important feature is the onboard dual-port RAM. This RAM can be accessed locally by the onboard processor without requiring any bus cycles and in addition can be accessed directly from the bus by another processor.

Onboard jumpers permitted us to adjust the memory and I/O configuration so that each processor occupies a full 1 Mbyte boundary on the 16 Mbyte (24-bit) address space. Consequently Chorus uses strictly linear addressing. An address map is shown in Fig. 1. Each board contains 128 kbytes of global RAM that begins at its megabyte boundary (0, 1, ..., F). Local (private) ROM starts at 0xFE0000 (hex) and continues through to 0xFEFFFF. Global I/O begins at 0xFF0000 and continues to 0xFFFDFF. Local I/O (serial ports, parallel ports, timer) runs from 0xFFFE00 to 0xFFFFFF.

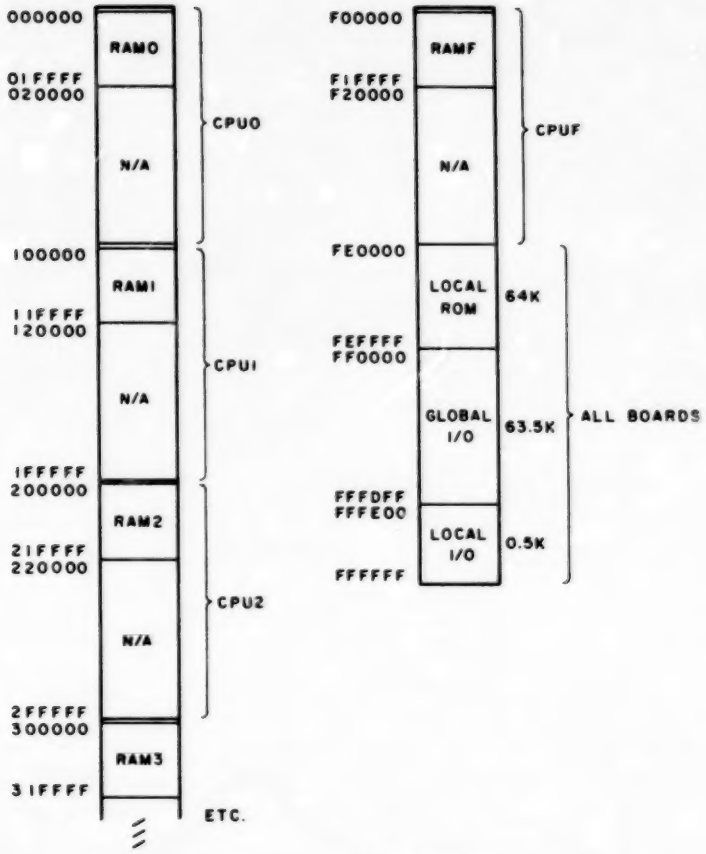

Figure 1. Memory and I/O Configuration

Figure 2 presents a generalized view of the Chorus system. This illustrates a system consisting of 16 processors, several I/O devices, and the Mailbox and Interprocessor Interrupt Generator. Also shown is the mechanism by which the interprocessor interrupts are transferred. Sixteen uncommitted bus lines on the "P2" portion of the IEEE 796 bus are used for this purpose.

Currently, each processor has a small amount of ROM-based code that is required during startup (cold start or restart). Processor 0, i.e., the processor whose address begins at 0x000000, contains a bootstrap loader that accepts Motorola S-records from the host VAX where programs are developed and stored. Processors 1 through F, which start up in a tight loop with no local RAM accesses, are downloaded first through the bootstrap loader. Last to be loaded is processor 0. When loaded, it automatically branches to a predetermined address and begins to establish its Harmony

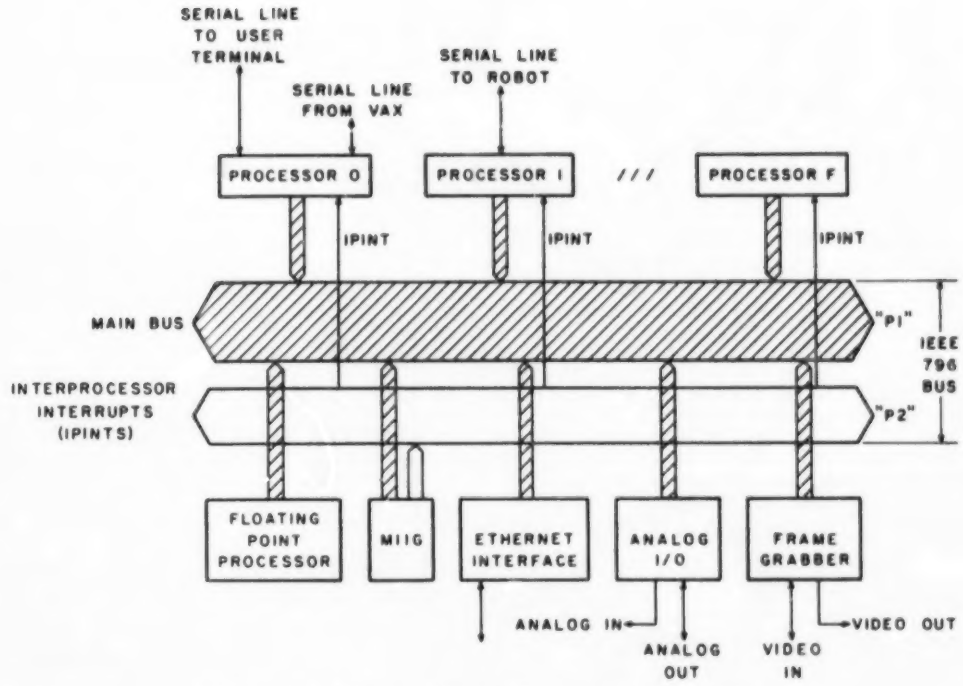

Figure 2. Generalized Chorus Configuration

environment. After it is finished, it "unlocks" the other processors and then waits for them to complete their initialization. The user program that begins to execute on processor 0 at the end of the initialization phase then must proceed to create all of the appropriate instances of tasks that are required by each of the other processors. Of course once a task starts to execute it can dynamically create and destroy other tasks on any processor, including its own, as required. (The current bootstrap loader permits a restart to occur without the need to reload by typing "g" at the user terminal after a system reset.)

CHAPTER 3. THE MAILBOX AND INTERPROCESSOR INTERRUPT GENERATOR (MIIG)

Introduction

Underlying the entire design of both Harmony and Chorus is the message-passing mechanism. This mechanism is used not only for transferring information between tasks but also for synchronizing two or more tasks. This is so because tasks are forced to block when messages are sent and received (1). Hardware support for this mechanism includes a separate mailbox for each of the 16 possible processors and an interprocessor

interrupt generator that is required in order to interrupt the destination processor at points in the protocol requiring action by the other processor, such as when a message has been sent. This hardware, called the MIIG, will be outlined in detail here.

To transfer a message, the originating task writes a task id to the appropriate mailbox and then generates an interrupt to the corresponding processor (i.e., the processor upon which the destination task is running). The processor receiving the interrupt, which may be the same processor upon which the originating task is running, clears the interrupt, transfers the task id from its mailbox to an internal queue, and then proceeds. The transfer of the message itself does not occur through the mailbox but rather is performed on a buffer-to-buffer basis.

From the preceding description, it is clear that some mechanism is required by which an interrupt can be directed to a specific processor. The IEEE 796 bus does not provide this functionality and so the design of an interrupt generator was necessary.

Message Passing With the MIIG

A task which wishes to send a message to another task must first determine which mailbox is to be used. Before it can actually send a message, the originator must check the mailbox to ensure that it is empty. Each mailbox consists of one long word (four consecutive bytes). When a mailbox is full the MSB is asserted. Therefore an originating task "spin-locks" on a full mailbox until the MSB is negated by the destination processor. (A "spin-lock" includes a test-and-set instruction and will not exceed a few instruction cycles.)

Once the MSB is found to be negated, the originating task writes four bytes of information (typically its task id) to the mailbox. This automatically asserts the MSB of the mailbox to indicate to other tasks that it is full (Fig. 3). Next the originator writes an all-zero word (16 bits) to the interrupt generator address that corresponds to the target processor. This causes an interprocessor interrupt to be generated which is seen only by the target processor. The target immediately clears the interrupt, transfers the information, and clears the mailbox thereby preparing it for the next message. (Further details concerning messages and their structure are best obtained from Harmony source code.)

Why a Separate Mailbox is Required

The OB68K1A single board computers used in Chorus were outlined earlier as having dual-port RAM. This means that an onboard arbiter decides on a cycle-by-cycle basis who is given access to the onboard RAM. It was also mentioned above that a task "spin-locks" on a busy mailbox. The "spin-lock"

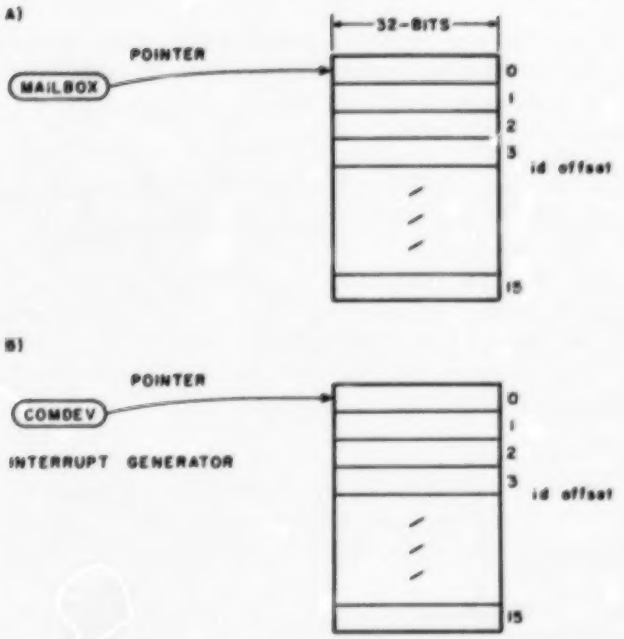

Figure 3. A software view of the mailbox and interprocessor interrupt generator. (a) Mailbox pointer + processor id = mailbox address. (b) Comdev pointer + processor id = processor interrupt address. Writing a 16-bit zero pattern into a processor interrupt address will generate an interrupt to the target processor.

includes a test-and-set instruction which must be indivisible to ensure reliability. Implementing the mailbox on one of the processors invites the possibilty that a local access may interfere with the indivisibility of a test-and-set instruction that is being executed by another processor via the bus. This possibility is eliminated by forcing all processors to access the mailbox in an identical manner over the bus.

Detailed Hardware Description

The mailbox and the interprocessor interrupt generator each occupy 64 bytes (16 long words each). For design convenience the two components have been combined such that together they occupy 128 bytes of contiguous memory with the interrupt generator occupying the lower 64 bytes. Switches are provided onboard so that the base address can be aligned to any 128 byte boundary. The resulting addresses (hex) are:

interrupt generator	XXXX00	to	XXXX3F
or	XXXX80	to	XXXXBF

current configuration	FDFF80	to	FDFFBF
mailbox	XXXX40	to	XXXX7F
or	XXXXC0	to	XXXXFF
current configuration	FDFFC0	to	FDFFFF

A more detailed address map is shown in Fig. 4.

In the following description, signals employing negative logic (i.e., those with bars in the circuit drawings) are followed by "-n".

Upper address bits ADR17-n to ADR7-n are decoded (Fig. 5) by a pair of comparators (Am25LS2521) and a single XOR gate to produce the board select (BRDSEL-n). When ADR6-n is negated, the interrupt generator is selected (INTSEL-n), and when asserted, the mailbox is selected (MBXSEL-n).

The interrupt generator is strobed (INTSTRB-n) when it is selected and a write command (MWTC-n) occurs. The mailbox is written to when it is selected and the write command occurs. A mailbox read occurs when it is selected and a read command (MRDC-n) occurs.

A transfer acknowledge (XACK-n) must be provided for any mailbox or interrupt generator access. Failure to do so will cause a bus error on the OB68K1A if the timeout generator is enabled. The interrupt generator is designed to respond to memory write cycles only. However, this design will also generate an acknowledge (INTACK-n) if an inadvertent read cycle happens to occur at an interrupt generator address. While this will have no affect upon the state of the interrupt generator, it prevents the occurrence of a bus error.

A write to any mailbox will automatically produce a correct (XACK-n). A read from a mailbox is slightly more complicated. Because the data fetched from the mailbox will not appear until some 70 ns after the MBXRD-n strobe is asserted, XACK-n has to be delayed. By shifting MBXRD-n through a 74F175 4-bit shift register, it is possible to select an appropriate delay (JP2). The 10 MHz asynchronous constant clock (CCLK-n) is used to clock the register. By selecting either BCCLK or BCCLK-n (JP1) it is possible to adjust the delay to the nearest 50 ns interval (a 100 ns delay is currently being used). The shift register is negated when MRDC-n is negated.

The mailbox itself is composed of a pair of 93419 RAMS (Fig. 6); one is associated with even addresses and the other with odd addresses. The

ADDRESS BIT PATTERN							HEX ADDRESS		SYMBOL
17	13	F	C	8	4	0	k=0	k=1	
aaaa	aaaa	aaaa	aaaa	k000	0000		XXXX00-03	(80-83)	SINT0
					0100		04-07	(84-87)	SINT1
					1000		08-0B	(88-8B)	SINT2
					1100		0C-0F	(BC-BF)	SINT3
				01	0000		10-13	(C0-C3)	SINT4
					0100		14-17	(C4-C7)	SINT5
					1000		18-1B	(C8-CB)	SINT6
					1100		1C-1F	(CC-CF)	SINT7
				10	0000		20-23	(D0-D3)	SINT8
					0100		24-27	(D4-D7)	SINT9
					1000		28-2B	(D8-DB)	SINTA
					1100		2C-2F	(DC-DF)	SINTB
				11	0000		30-33	(E0-E3)	SINTC
					0100		34-37	(E4-E7)	SINTD
					1000		38-3B	(E8-EB)	SINTE
					1100		3C-3F	(EC-EF)	SINTF
aaaa	aaaa	aaaa	aaaa	k100	0000		XXXX40-43	(C0-C3)	MBX0
					0100		44-47	(C4-C7)	MBX1
					1000		48-4B	(C8-CB)	MBX2
					1100		4C-4F	(CC-CF)	MBX3
				101	0000		50-53	(D0-D3)	MBX4
					0100		54-57	(D4-D7)	MBX5
					1000		58-5B	(D8-DB)	MBX6
					1100		5C-5F	(DC-DF)	MBX7
				110	0000		60-63	(E0-E3)	MBX8
					0100		64-67	(E4-E7)	MBX9
					1000		68-6B	(E8-EB)	MBXA
					1100		6C-6F	(EC-EF)	MBXB
				111	0000		70-73	(F0-F3)	MBXC
					0100		74-77	(F4-F7)	MBXD
					1000		78-7B	(F8-FB)	MBXE
					1100		7C-7F	(FC-FF)	MBXF

Figure 4. Address map of the MIIG. "a" and "k" are selected from onboard switches

corresponding chip selects, CSE-n and CSO-n, are generated as shown in Table 1 (Fig. 5).

The data path that precedes the RAM's contains what is politely called "byte swapping logic". As far as the 68000 is concerned, even addresses contain high bytes while odd addresses contain low bytes. The original design of the IEEE 796 bus (i.e., the Multibus) assumed otherwise. To

conform to the bus standard, during a 16-bit (word) transfer, the OB68K1A board swaps the high byte with the low byte and therefore the mailbox board must do the same. In addition, all single byte transfers take place across the low byte (DAT0-n to DAT7-n) on the bus. The resulting chip selects are summarized in Table 2 (Fig. 5).

It is of no consequence that the logic "free wheels"; i.e. is not selected by MBXSEL.

During a write cycle, the output buffers downstream from the RAM's are disabled so as not to collide with the byte swapper multiplexers. During a read cycle the byte swappers reverse direction while the RAM output buffers are enabled (MBXRD-n).

The interrupt generator (Fig. 7) consists of a pair of octal addressable latches (74LS259). The appropriate latch is selected by addresses BADR2 - 5. The ZERO input determines whether or not the addressed latch will be set. An all-zero bit pattern written to the interrupt generator is detected by the zero detect logic which sets the ZERO line accordingly. Drivers are provided for each of the 16 outputs. These are routed to the "P2" connector on the card edge.

Detailed timing diagrams are presented in Figs. 8 and 9.

Table 1. Mailbox RAM chip selects

MBXSEL	BBHEN-n	BADR0-n	CSE-n	CSO-n	
F	X	X	F	F	Not selected
T	F	F	T	F	Even byte
T	F	T	F	T	Odd byte
T	T	F	T	T	Word
T	T	T	-	-	Never occurs

Table 2. Mailbox datapath selects

BBHEN-n	BADR0	ODDSEL-n	EVSEL-n	WRDSEL-n	
F	F	F	T	F	Even byte
F	T	T	F	F	Odd byte
T	F	F	T	T	Word
T	T	F	F	F	Never

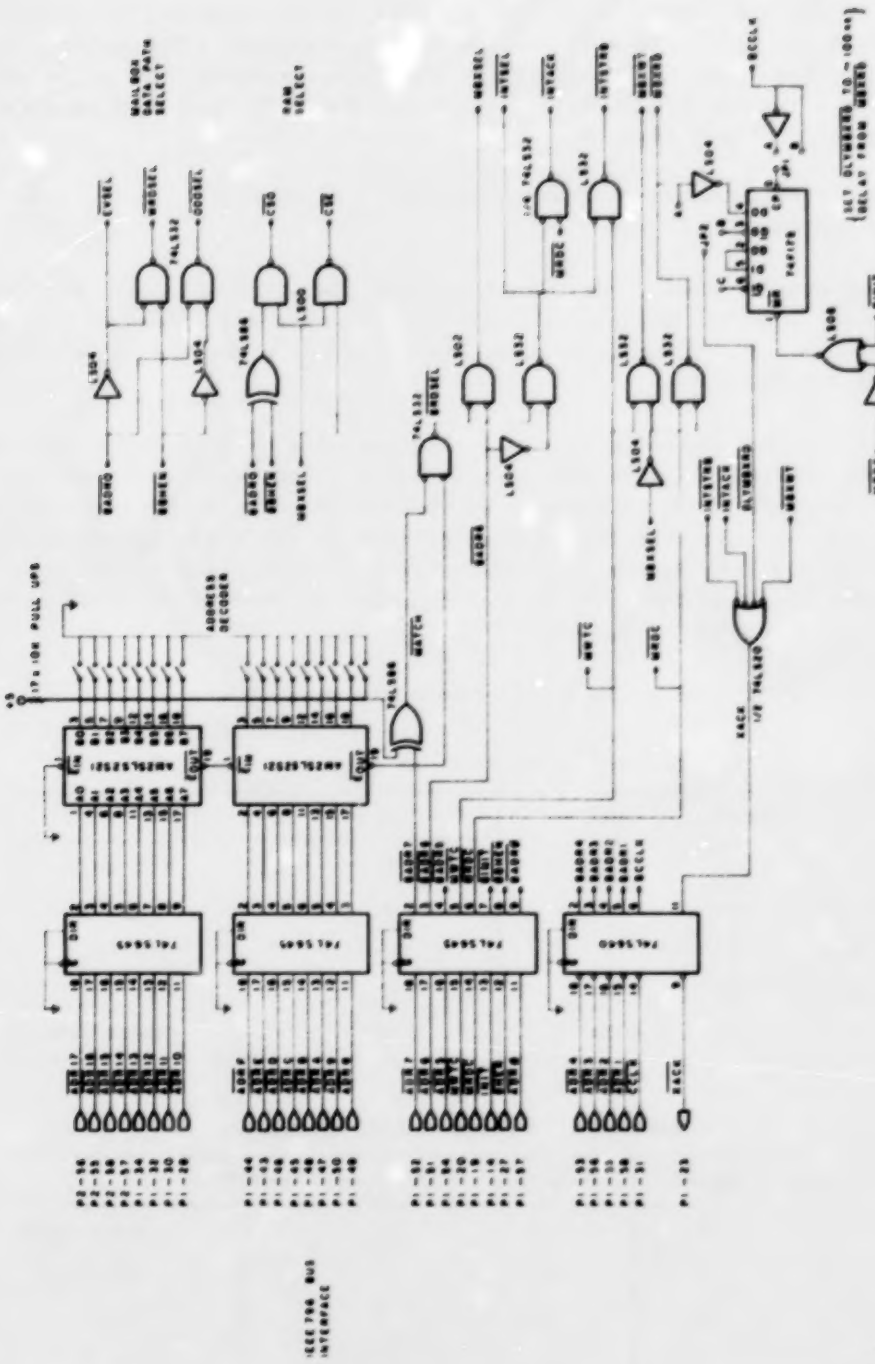

Figure 5. MIG design - IEEE 796 bus interface and control logic

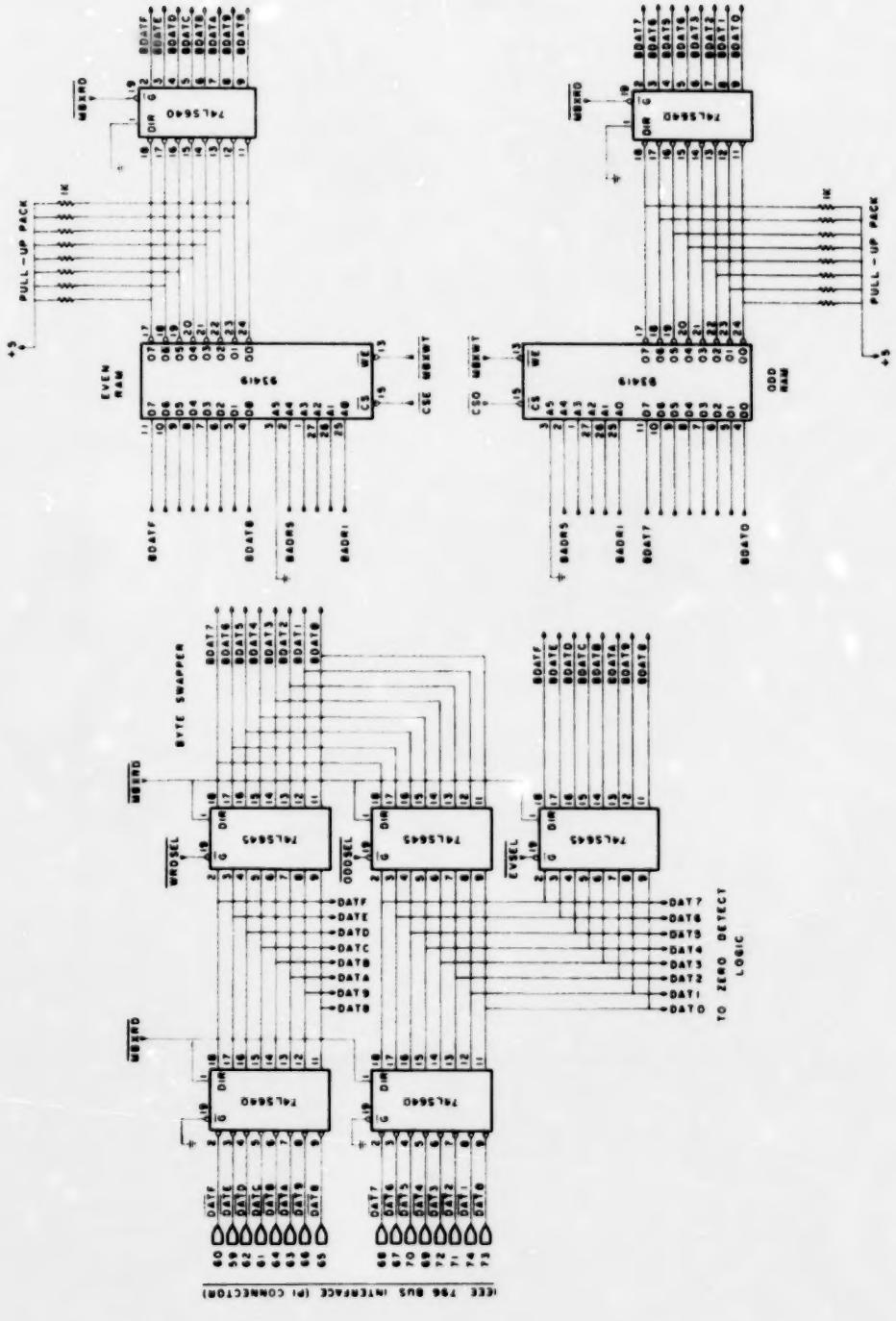

Figure 6. MIG design - mailbox and data paths

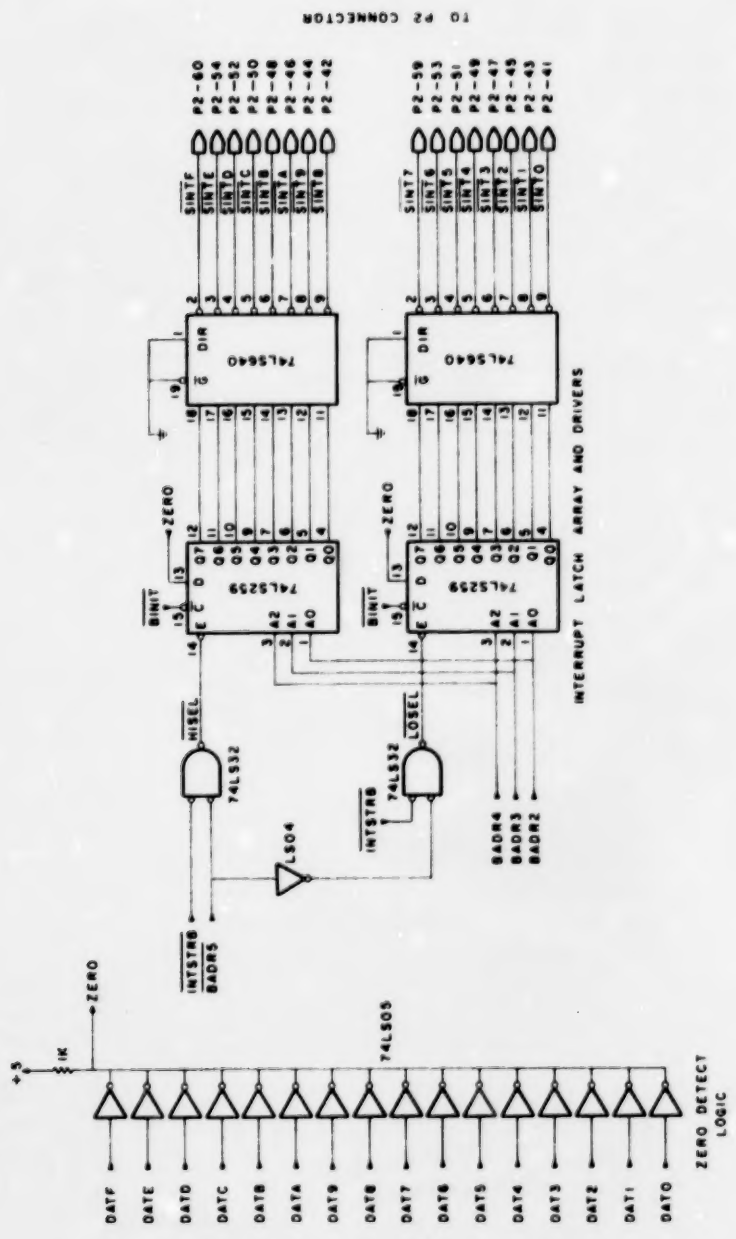

Figure 7. MIIG design - interprocessor interrupt generator

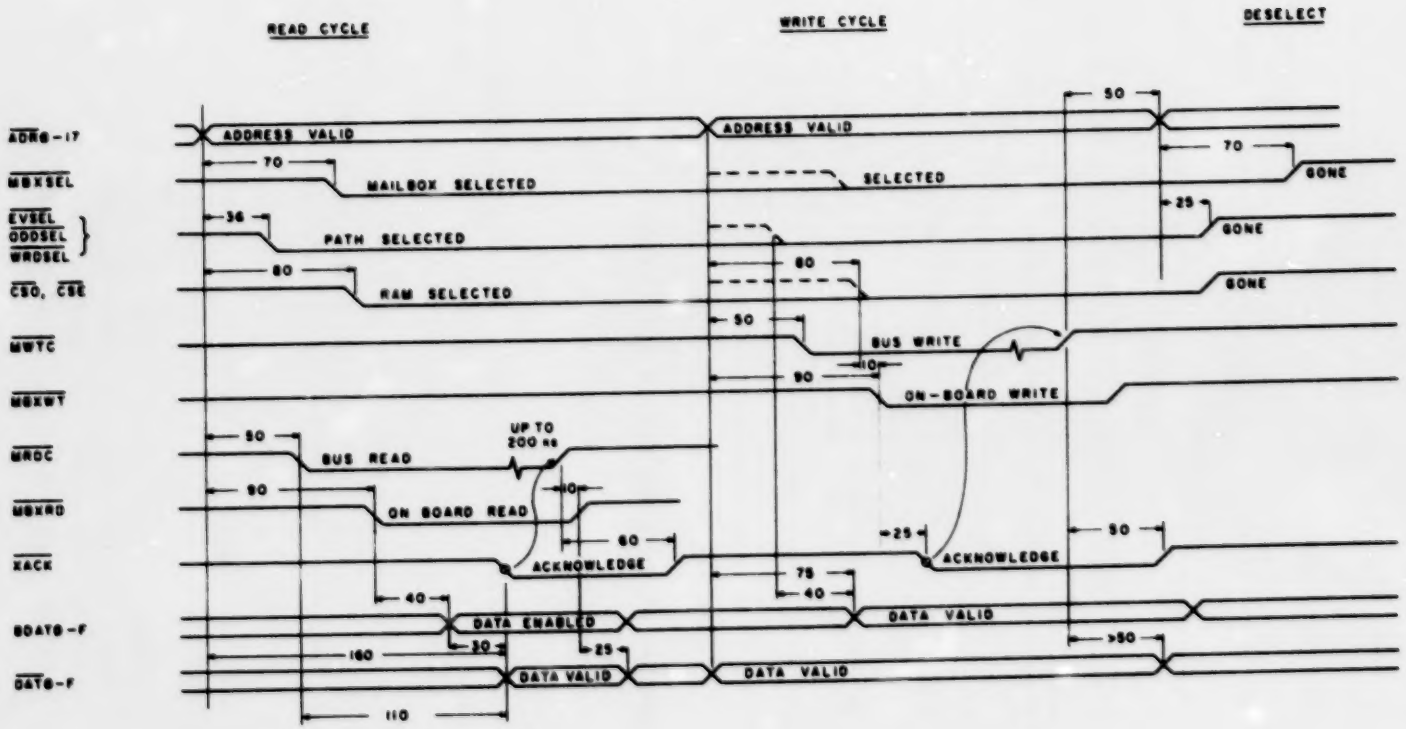

Figure 8. Mailbox timing

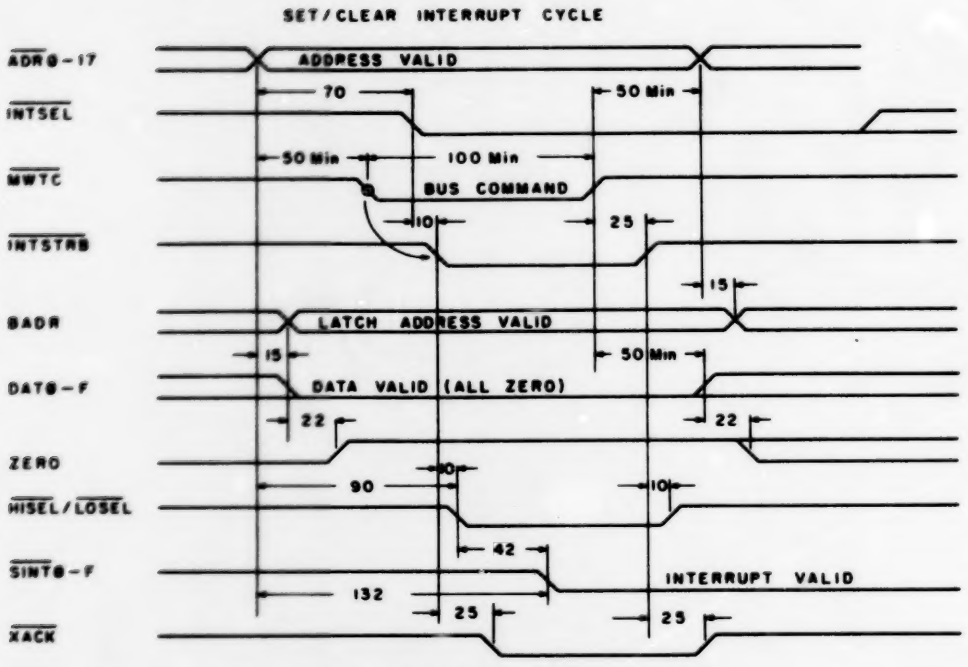

Figure 9. Interrupt generator timing

CHAPTER 4. BUS AND CHASIS CONFIGURATION DETAILS

This chapter addresses some of the implementation details associated with Chorus.

Parallel priority resolution or PPR is essential in this system. Most vendors of card cages for the IEEE 796 bus support PPR. It has been implemented as described in the standard (4).

Address lines ADR14-n to ADR17-n on the "P2" connector required 2.2 kΩ pullup resistors to +5 V for proper termination. The interprocessor interrupt lines SINT0-n to SINTF-n had to be terminated with 2.2 kΩ pullups as well.

Interprocessor interrupts are carried on uncommitted lines associated with the "P2" connector:

SINT0-n	P2 - 41	SINT1-n	P2 - 43
SINT2-n	P2 - 45	SINT3-n	P2 - 47
SINT4-n	P2 - 49	SINT5-n	P2 - 51
SINT6-n	P2 - 53	SINT7-n	P2 - 59

SINT8-n	P2 - 42	SINT9-n	P2 - 44
SINTA-n	P2 - 46	SINTB-n	P2 - 48
SINTC-n	P2 - 50	SINTD-n	P2 - 52
SINTE-n	P2 - 54	SINTF-n	P2 - 60

Each processor has its own interprocessor interrupt which must be physically connected to interrupt level 6 on the OB68K1A. It was decided that the interconnection from the SINTx-n line on the "P2" backplane to the appropriate pin on the OB68K1A would be made by reallocating the INT0-n line (P1 - 41). Consequently the INT0-n line on the backplane had to be isolated at each processor location (Fig. 10.). Then at each processor slot on the backplane, a jumper connected a SINTx-n line to the "unbussed" INT0-n line. Finally, a jumper on the OB68K1A connected the interprocessor interrupt to the level 6 interrupt input.

CHAPTER 5. CONCLUSIONS

To date, a Chorus machine consisting of two processors has been successfully demonstrated in a vision-based robot system. It is our intention to extend the application further, with additional sensors and processors, in order to more fully evaluate the performance of the Harmony/Chorus "duet".

New single board computers are being announced on a regular basis and some of these promise to support multiprocessing, mainly with dual-port RAM. It is likely that in the future Chorus will migrate onto other busses and employ a mixture of different processors. There is nothing intrinsic about Harmony that should prevent this from happening. The use of a

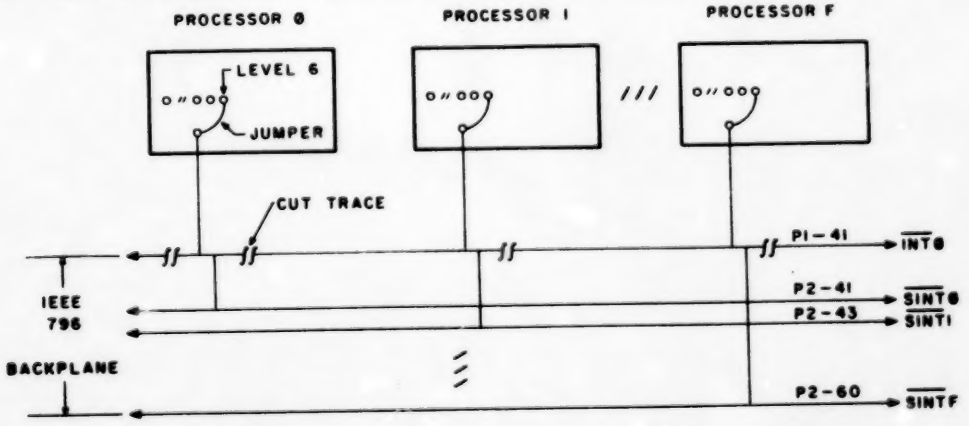

Figure 10. Interconnection of interprocessor interrupts on the backplane (pull up resistors are not shown)

separate MIIG board facilitates this by predetermining the mailbox and interprocessor interrupt structure.

There is no doubt that modular systems that exploit parallelism as easily as does Chorus will be of considerable interest in the future.

REFERENCES

1. W.M. Gentleman. Using the Harmony Operating System. NRC/ERB-966. Division of Electrical Engineering, National Research Council of Canada, Ottawa, Ont. December 1983.

2. The Multibus Buyer's Guide. Ironoak Company, La Jolla, CA. 1983.

3. OB68K1A MC68000 Single Board Computer User's Manual. Omnibyte Corp, West Chicago, IL. 1983.

4. IEEE STANDARDS #796 - 1983 Microcomputer System Bus. IEEE Press, New York, NY. 1983.

POOR

QUALITY

ORIGINAL